Tutorial On Converting Frequency Response From Cadence to LTI Model in Matlab

In Cadence, we can easily simulate the frequency response of a circuit, i.e., the magnitude and phase as function of frequency. This effectively describes the transfer function of the system under consideration. However, it is not easy to extract the pole/zero locations once the system has more than two poles. The pole/zero location of the system becomes important when we consider a system under feedback. Specifically, we need to know the pole/zero locations in order to effectively compensate the system for a given phase margin requirement. To do this, we will use Matlab. This tutorial shows the step by step procedure to convert Frequency Response plot from Cadence to Matlab.

Step 1. Start by building a circuit in Cadence. In this example, it is a simple Common-Source amplifier.

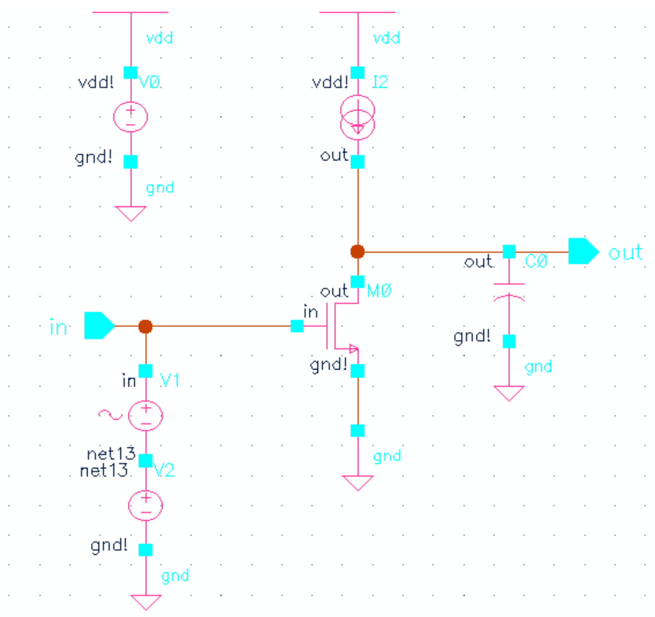
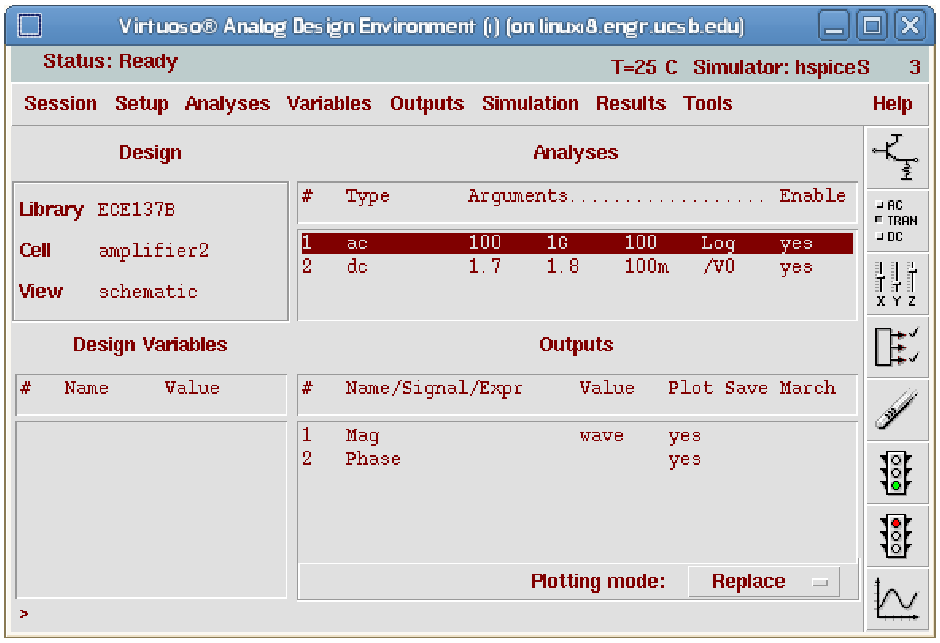


Figure 1: Common-source amplifier

Step 2. Run AC simulation in Analog Design Environment, sweep frequency of interest. In this example, it is from 100 Hz to 1GHz, with 100 sweep points per decade of frequency.



Status: Ready T=25 C Simulator: hspiceS 3

Design

Library	ECE137B
Cell	amplifier2
View	schematic

Analyses

#	Type	Arguments				Enable
1	ac	100	1G	100	Log	yes
2	dc	1.7	1.8	100m	/V0	yes

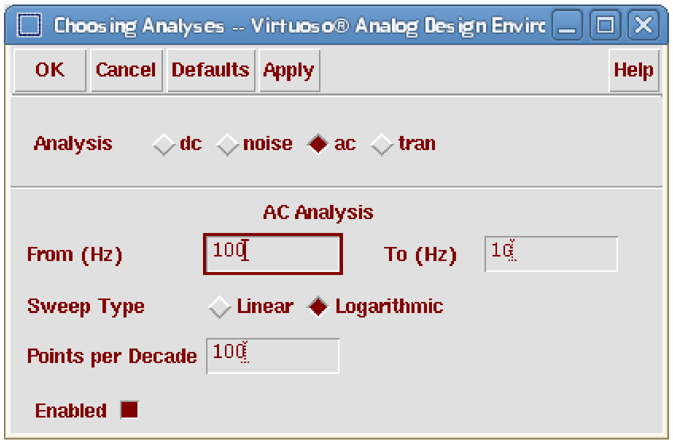
Design Variables

#	Name	Value

Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	Mag	wave	yes		
2	Phase		yes		

Figure 2: Analog Design Environment



Analysis: dc, noise, [✓ ac], tran

AC Analysis

From (Hz) 100 To (Hz) 1G

Sweep Type: Linear [✓ Logarithmic]

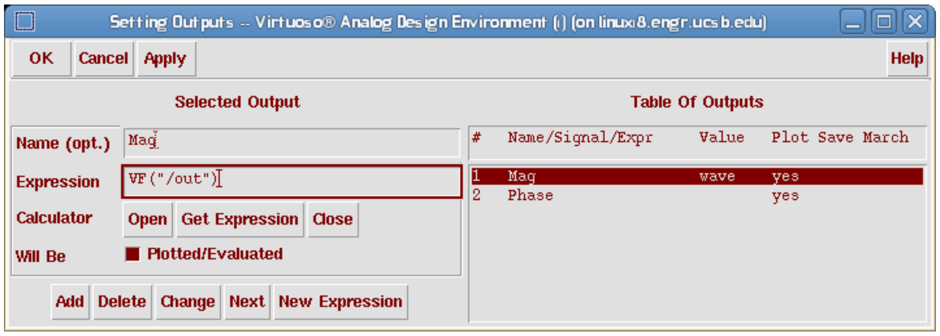
Points per Decade 100

Enabled ■

Figure 3: AC simulation setup

Step 3. Define outputs to be plotted. i.e. Magnitude and Phase of the output node. Note that in this example, the input is set at 1VAC, so the output node voltage is effectively Vout/Vin of the system

- Define an output called Mag with the expression *VF("/out")*, where /out is the node name of interests.
- Define an output called Phase with the expression *phase(VF("/out"))*



Setting Outputs -- Virtuoso® Analog Design Environment (i) (on linux8.engr.ucsb.edu)

OK Cancel Apply Help

Selected Output

Name (opt.)	Mag
Expression	VF("/out")
Calculator	Open · Get Expression · Close
Will Be	■ Plotted/Evaluated

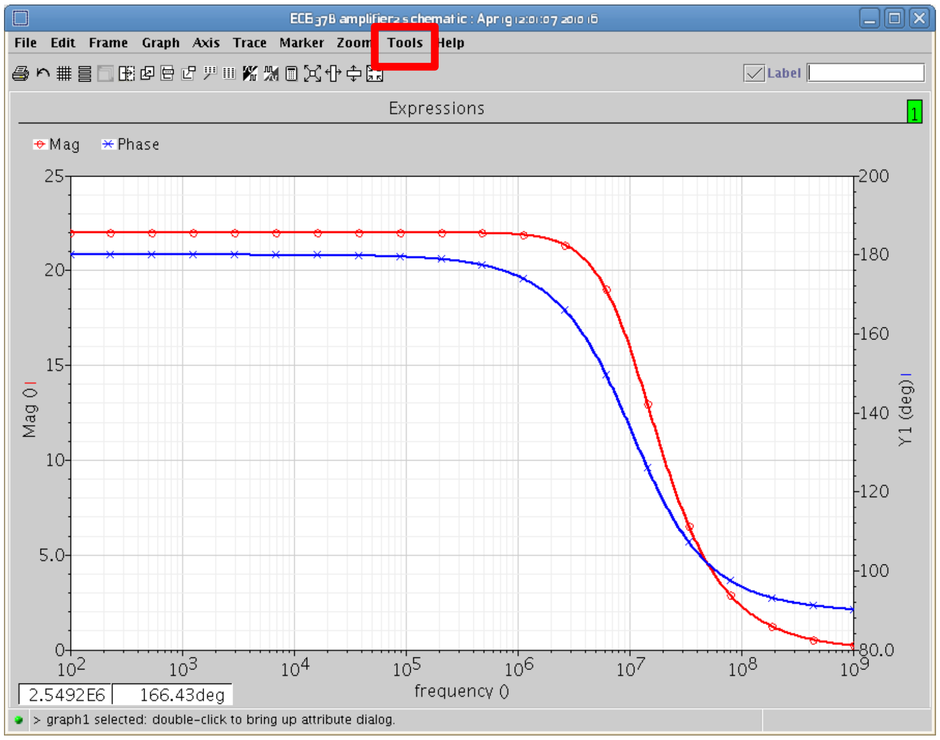
Add Delete Change Next New Expression

Table Of Outputs

#	Name/Signal/Expr	Value	Plot	Save	March
1	Mag	wave	yes		
2	Phase		yes		

Figure 4: Output setup window

Step 4. Simulate and plot outputs

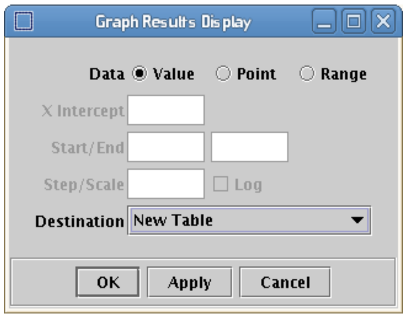


> graph1 selected: double-click to bring up attribute dialog.

Figure 5: Frequency response

Step 5. Generate table of magnitude/phase response data.

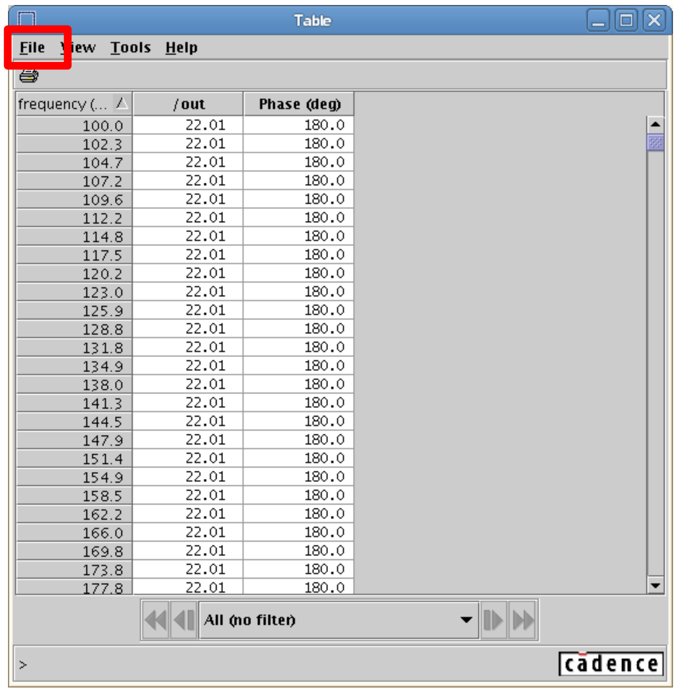
- Select the magnitude response curve on the plot window. Go to *Tools* menu, Select *Table*. Choose *New Table* as Destination. Click *OK*.



Data [✓ Value] Point Range
X Intercept
Start/End
Step/Scale ☐ Log
Destination New Table

Figure 6: Table subwindow

- With the table open, go back to the plot window, and select the phase response curve. Go to *Tools* menu, select *Table* again. Choose *Append* as Destination.



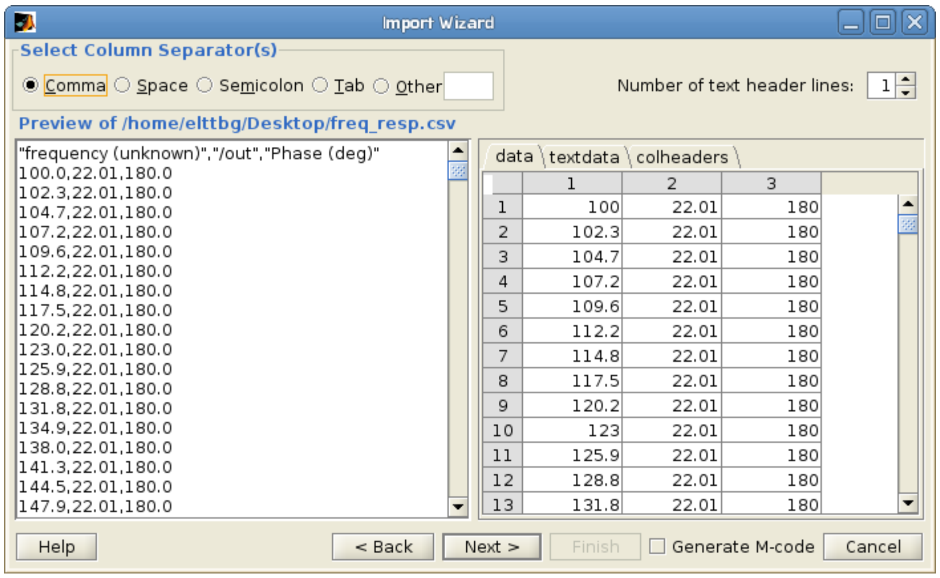
frequency (...	/out	Phase (deg)
100.0	22.01	180.0
102.3	22.01	180.0
104.7	22.01	180.0
107.2	22.01	180.0
109.6	22.01	180.0
112.2	22.01	180.0
114.8	22.01	180.0
117.5	22.01	180.0
120.2	22.01	180.0
123.0	22.01	180.0
125.9	22.01	180.0
128.8	22.01	180.0
131.8	22.01	180.0
134.9	22.01	180.0
138.0	22.01	180.0
141.3	22.01	180.0
144.5	22.01	180.0
147.9	22.01	180.0
151.4	22.01	180.0
154.9	22.01	180.0
158.5	22.01	180.0
162.2	22.01	180.0
166.0	22.01	180.0
169.8	22.01	180.0
173.8	22.01	180.0
177.8	22.01	180.0

Figure 7: Magnitude/Phase data table

- Now you should have a similar table as shown in Fig. 7.

Step 6. Save table as CSV file by selecting the *File* menu, *Save as CSV*. Now we are ready to import the data file into Matlab

Step 7. Launch Matlab, import the CSV file you have created by selecting *File*, *Import data* on Matlab's main window. Choose *Comma* as column separator, click *Next*, then *Finish*. You should see three variables on Matlab's workspace, *colheaders*, *data* and *textdata*. The variable of interest is data, which should have 3 columns, col. 1 corresponds to the frequency, col. 2 the magnitude and col. 3 the phase.



Select Column Separator(s): [✓ Comma] Space Semicolon Tab Other

Number of text header lines: 1

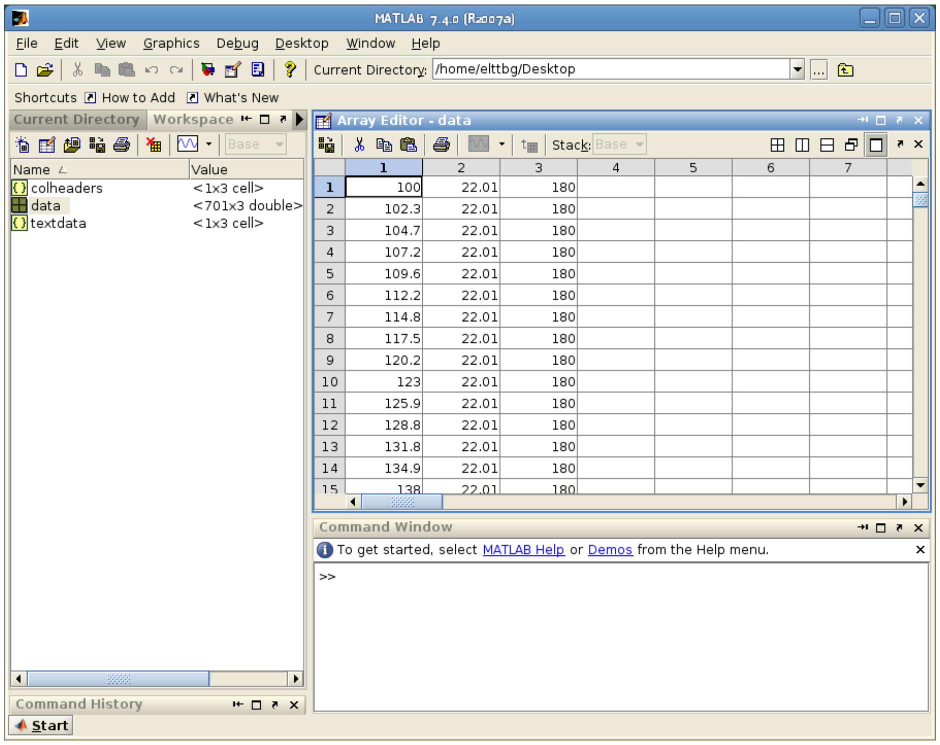
Preview of /home/elttbg/Desktop/freq_resp.csv

```
"frequency (unknown)","/out","Phase (deg)"
100.0,22.01,180.0
102.3,22.01,180.0
104.7,22.01,180.0
107.2,22.01,180.0
109.6,22.01,180.0
112.2,22.01,180.0
114.8,22.01,180.0
117.5,22.01,180.0
120.2,22.01,180.0
123.0,22.01,180.0
125.9,22.01,180.0
128.8,22.01,180.0
131.8,22.01,180.0
134.9,22.01,180.0
138.0,22.01,180.0
141.3,22.01,180.0
144.5,22.01,180.0
147.9,22.01,180.0
```

data | textdata | colheaders

	1	2	3
1	100	22.01	180
2	102.3	22.01	180
3	104.7	22.01	180
4	107.2	22.01	180
5	109.6	22.01	180
6	112.2	22.01	180
7	114.8	22.01	180
8	117.5	22.01	180
9	120.2	22.01	180
10	123	22.01	180
11	125.9	22.01	180
12	128.8	22.01	180
13	131.8	22.01	180

Figure 8: Matlab import wizard window



Name	Value
colheaders	<1x3 cell>
data	<701x3 double>
textdata	<1x3 cell>

	1	2	3	4	5	6	7
1	100	22.01	180				
2	102.3	22.01	180				
3	104.7	22.01	180				
4	107.2	22.01	180				
5	109.6	22.01	180				
6	112.2	22.01	180				
7	114.8	22.01	180				
8	117.5	22.01	180				
9	120.2	22.01	180				
10	123	22.01	180				
11	125.9	22.01	180				
12	128.8	22.01	180				
13	131.8	22.01	180				
14	134.9	22.01	180				
15	138	22.01	180				

```
To get started, select MATLAB Help or Demos from the Help menu.
>>
```

Figure 9: Matlab main window

Step 8, create complex dataset using the following Matlab command:

*cdata=[data(:,1),data(:,2).*exp(j*data(:,3)*pi/180)];*

What we are doing here is convert the magnitude/phase information into a complex number in the form of $mag \times \exp[j \times phase \frac{\pi}{180}]$.

Step 9. Convert the complex dataset *cdata* to Frequency Response Data model using the following command:

*sys=frd(cdata(:,2),cdata(:,1)*2*pi)*

Step 10. Convert FRD model to LTI model using the following command:

LTIsys=pem(sys)

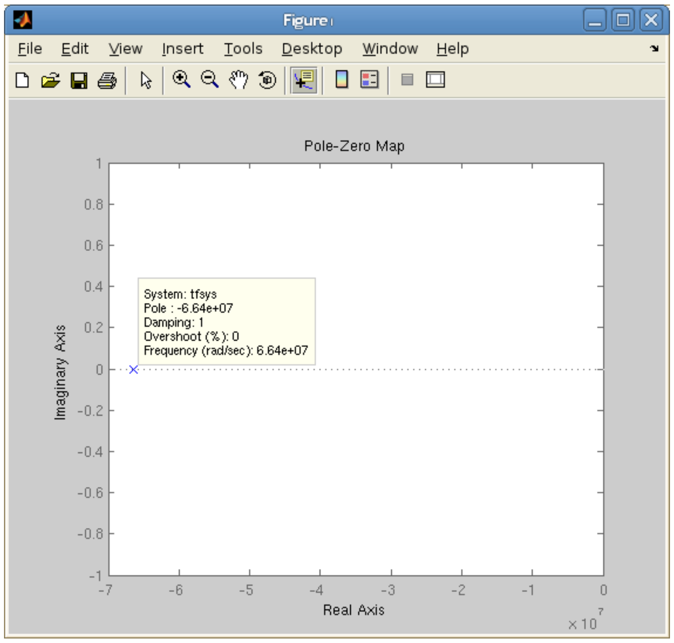
tfsys=tf(LTIsys)

pem() outputs a state-space representation of the system, *tf()* converts the state-space system to the famliar polynomial H(s)=N(s)/D(s) form.

For more information regarding *frd* and *pem*, read Matlab's help file under *frd* and *pem*

Step 11. Finally, we can visualize the pole/zero locations on the complex plain by using the command:

pzmap(tfsys)



```
System: tfsys
Pole : -6.64e+07
Damping: 1
Overshoot (%): 0
Frequency (rad/sec): 6.64e+07
```

Figure 10: pole/zero map

As expected, the example system has a single pole located at 66.7 Mrad/s or 10.56MHz , which corresponds to the 3dB bandwidth of the system.

Alternatively, we can also look at the Bode plot by using the command

bode(tfsys)

which should replicate what you have simulated in Cadence

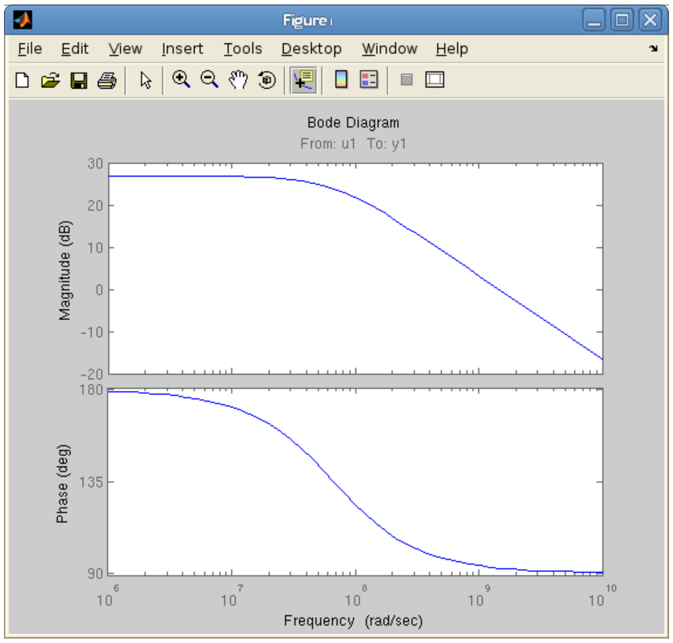


Figure 11: Bode plot

In the next tutorial, we will show you how to use SISOtools to manipulate the pole/zero locations when we put the system in feedback.